Министерство науки и образования Российской Федерации

ФГБОУ ВО «Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Лабораторная работа №3**

Дисциплина: «Системы искусственного интеллекта»

«Базы знаний и онтологии»

Вариант 1

Выполнил студент  
группы ИВТАCбд-41  
Абросимов К. С.

Проверил:  
преподаватель кафедры ВТ  
Хайруллин И. Д.

Ульяновск, 2024

**Постановка задачи**

В рамках данной лабораторной работы необходимо разработать систему управления для автоматизации заданного технического объекта. Для хранения базы правил и онтологии предметной области предлагается использовать Neo4j, либо любое другое хранилище на выбор (реляционная СУБД, Protege). Задание предполагает проектирование логической модели управления, построение и настройку правил управления, а также разработку симулятора для проверки работы системы управления. Для формирования условий срабатывания правил необходимо использовать фаззификацию на основе нечеткой логики, для формирования управляющих инструкций - дефаззификацию.

Также необходимо разработать минимально рабочий симулятор предметной области с дискретным программным управлением.

**Ход работы**

Вариант работы. Автоматизация полива растений в теплице: разработать систему управления поливом в зависимости от температуры, влажности, типа растений и времени суток.

1. Ознакомиться с особенностями выбранного объекта проектирования, включая его функциональные требования, ограничения и задачи автоматизации.

Растения будут иметь полный цикл роста от начала посадки до точки полного выращивания, когда программа перестаёт наблюдать за растением.

Растения должны быть сгруппированы по типу, что влечёт за собой написание правил полива под каждый тип: одним растениям нужно мало влаги, другим – много.

Время суток – должна производиться ротация дня по временам суток (утро, день, вечер, ночь).

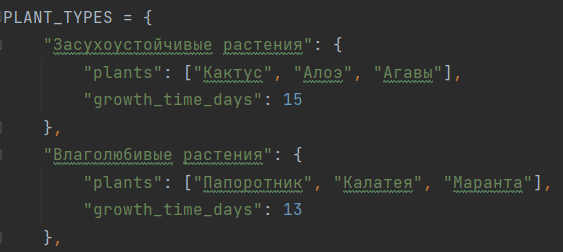
Температура – случайно генерируемая величина. Будет случайно генерироваться каждую смену времени суток.

Влажность – основная величина, которая влияет на жизнеспособность растений. Влажность должна поддерживаться в определённом диапазоне для определённых растений и будет уменьшаться в зависимости от температуры.

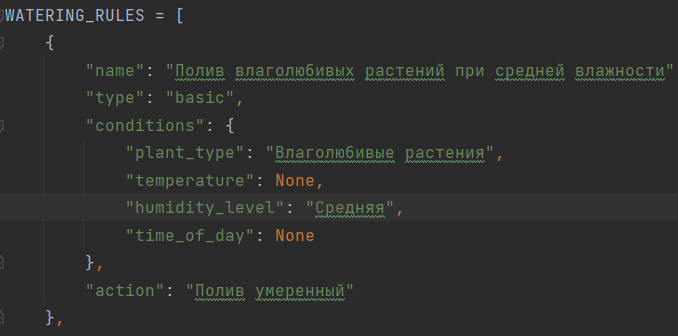
1. Разработка базы данных знаний.

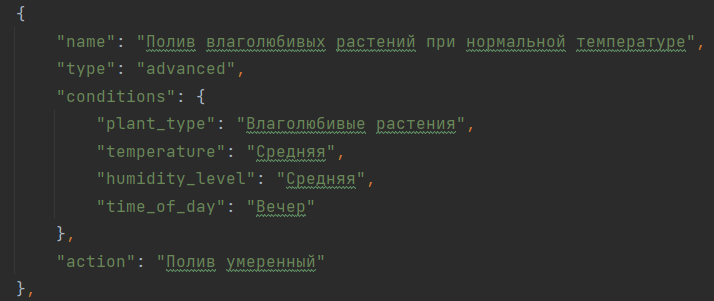
В онтологии предметной области будут курировать две основные переменные:

* Растения. База данных будет хранить названия растения по типам. У каждого типа будет своё время роста. Нужно для определения правил полива.



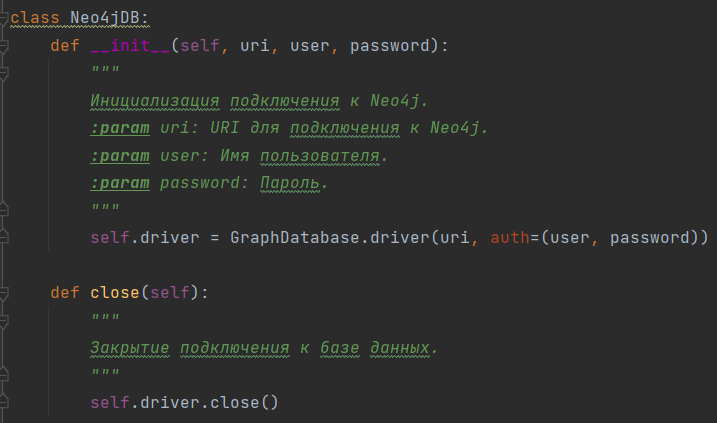
* Правила полива. База данных будет хранить условия (тип растения, время суток, температура, влажность) и значение обильности полива. Так же правила будут разделены на две категории: базовые (полив обычный) и продвинутые (полив при особенных условиях).





Для создания базы данных используется Neo4j, запросы будут проводиться на языке Cypher.

Для начала нужно создать файл knowledge\_base.py, который будет хранить класс Neo4j:



Одни методы этого класса будут заносить все данные массивов в базу данных, другие будут проводить чтение данных.

Следующие две функции заносят данные в базу данных, они используются лишь один раз (или же если появились изменения в самих массивах правил и растений).

* Создание правил

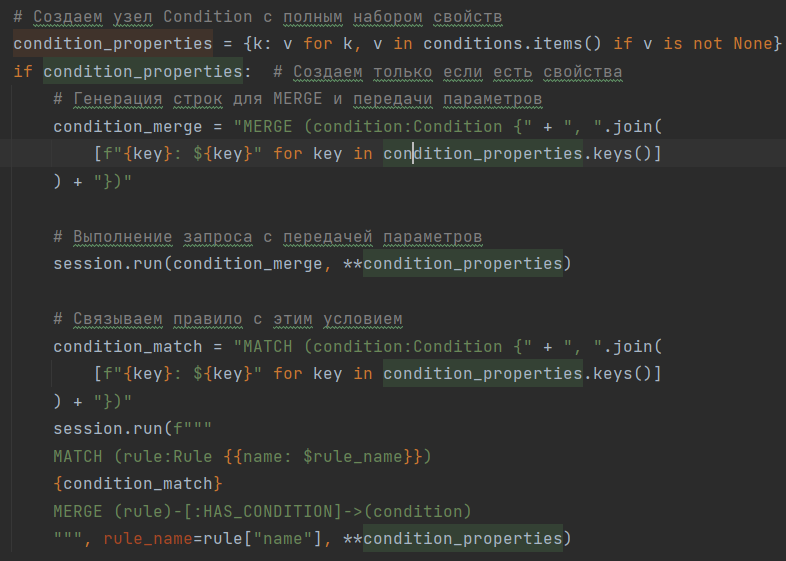
Программа выполняет запрос

MATCH (n) DETACH DELETE n

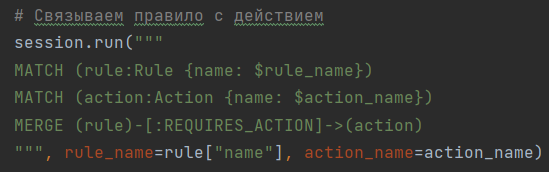
Для полной очистки базы данных.

MERGE (action:Action {name: $action\_name})  
MERGE (rule:Rule {name: $rule\_name, type: $rule\_type})

Перебирая каждое правило отдельно, мы сначала создаём действие и правило как отдельные узлы.



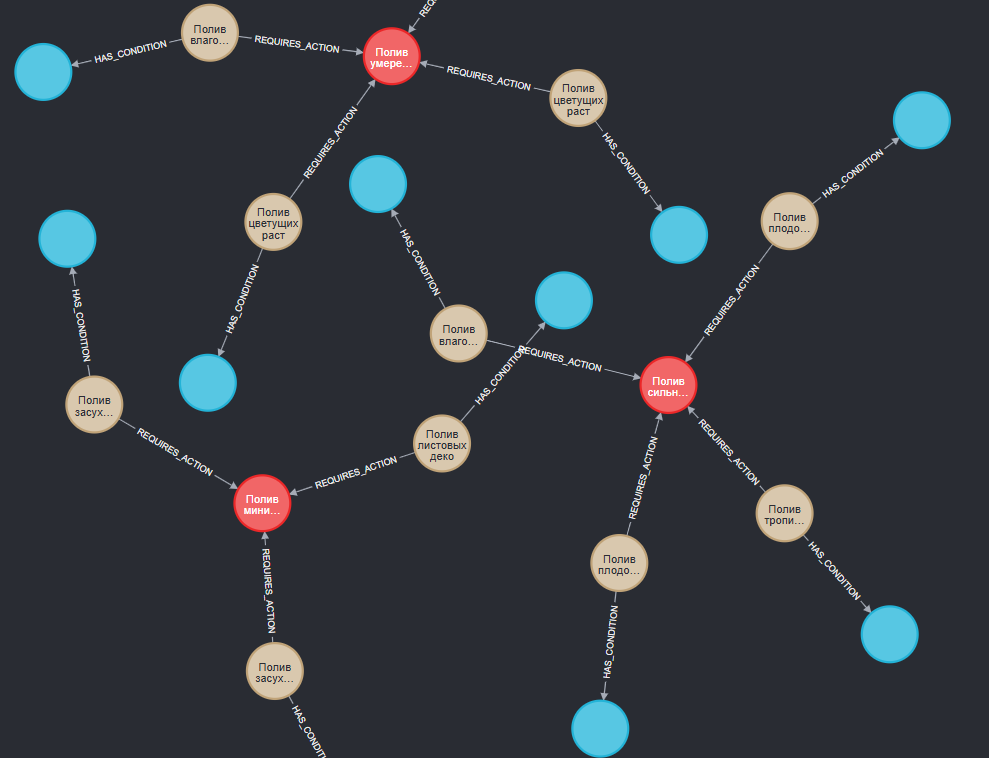
Далее программа создаёт узел «Условие» и задаёт в него условия, объединяя с правилом связью HAS\_CONDITION.



Напоследок проводится связь между правилом и последующим действием.

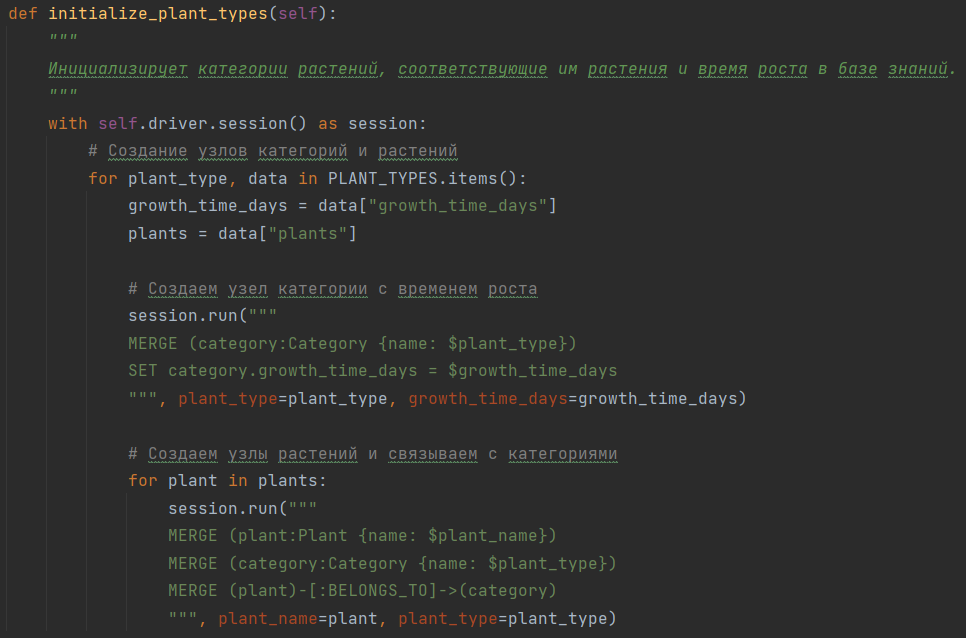
Запрос MATCH p=()-->() RETURN p LIMIT 25

будет иметь вид:



Синий – условие, русый – правило, красный – действие.

* Создание типов растений и растений



Функция сохранения типов растений с названиями в базу данных.

Сначала мы создаём узлы категорий – типы растения, затем ставим свойство «время роста»:

MERGE (category:Category {name: $plant\_type})  
SET category.growth\_time\_days = $growth\_time\_days

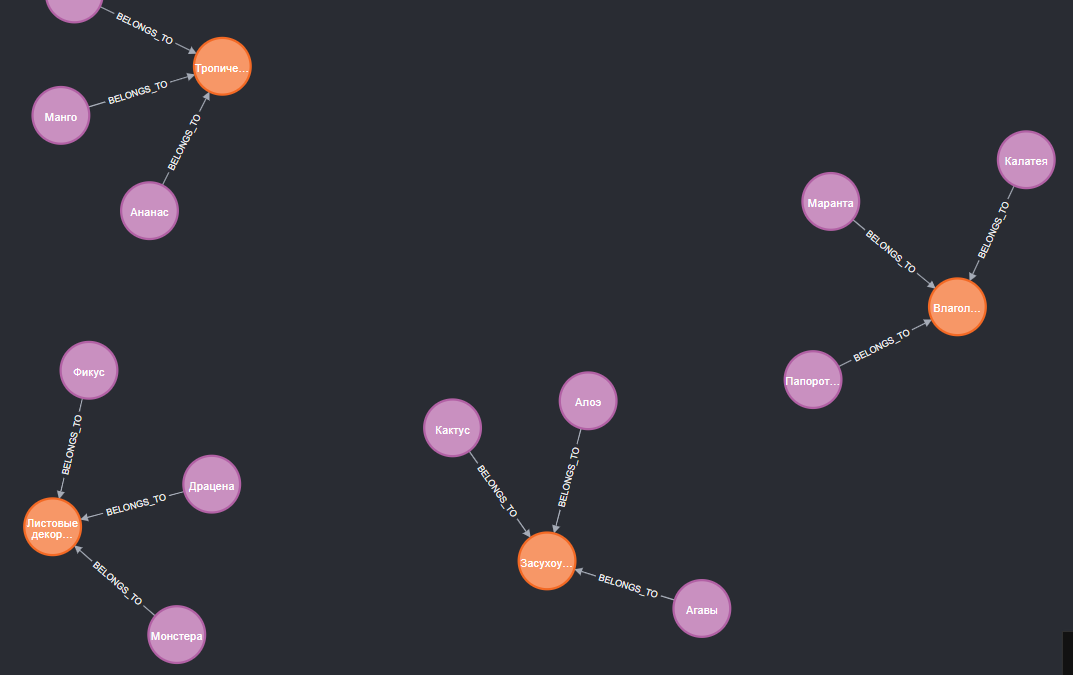
Затем мы создаём узел растения, проверяем существование типа растения в базе, а потом связываем с типом:

MERGE (plant:Plant {name: $plant\_name})  
MERGE (category:Category {name: $plant\_type})  
MERGE (plant)-[:BELONGS\_TO]->(category)

В Neo4j по запросу

MATCH p=()-[r:BELONGS\_TO]->() RETURN p LIMIT 25

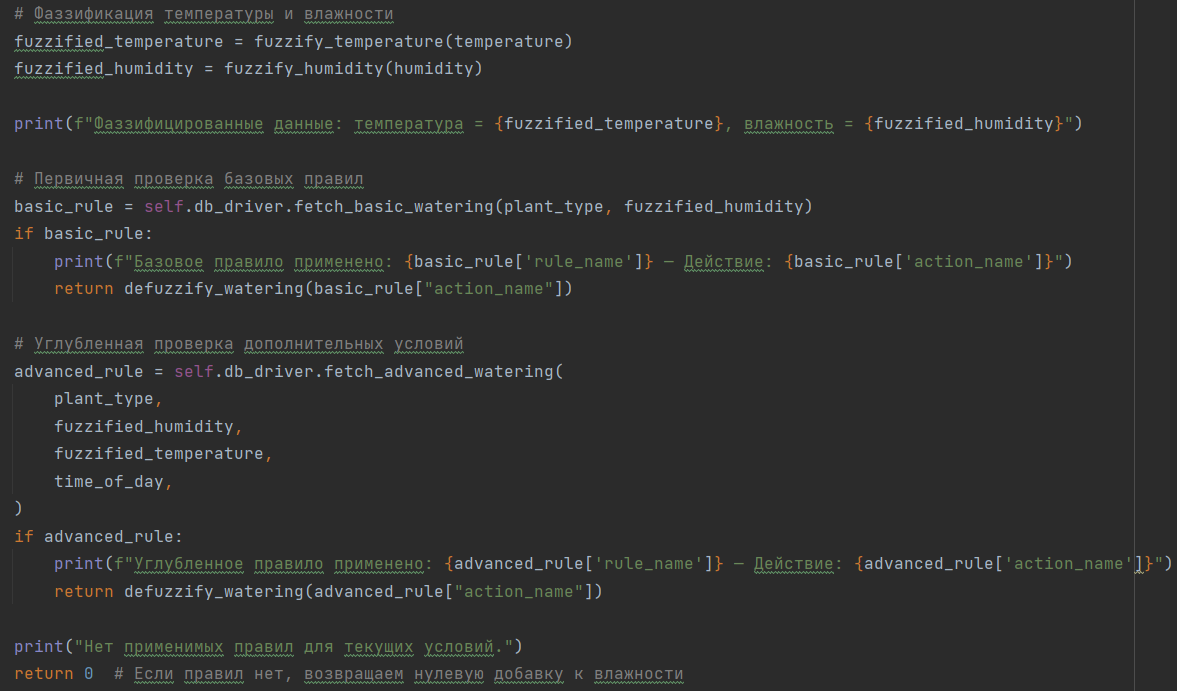
выводится следующий граф:

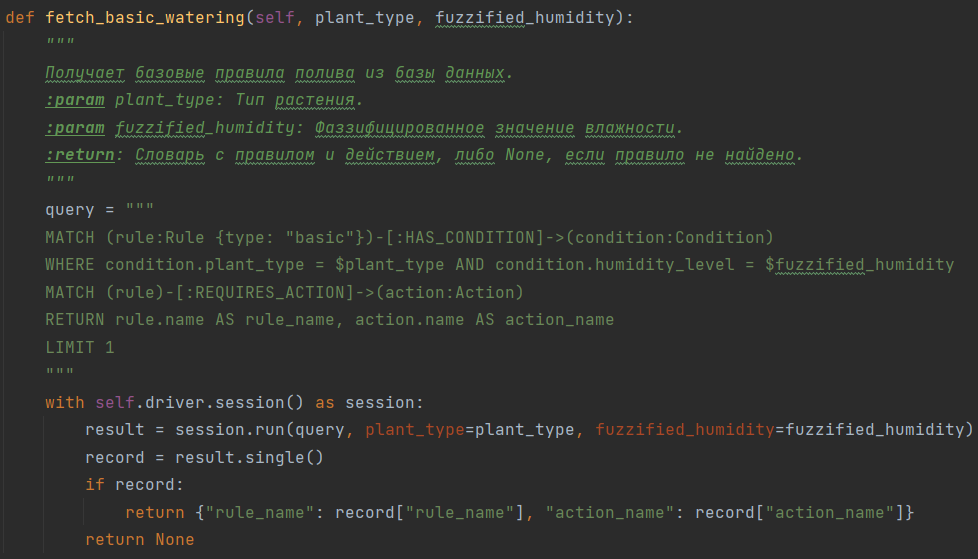


1. Проектирование логической модели управления

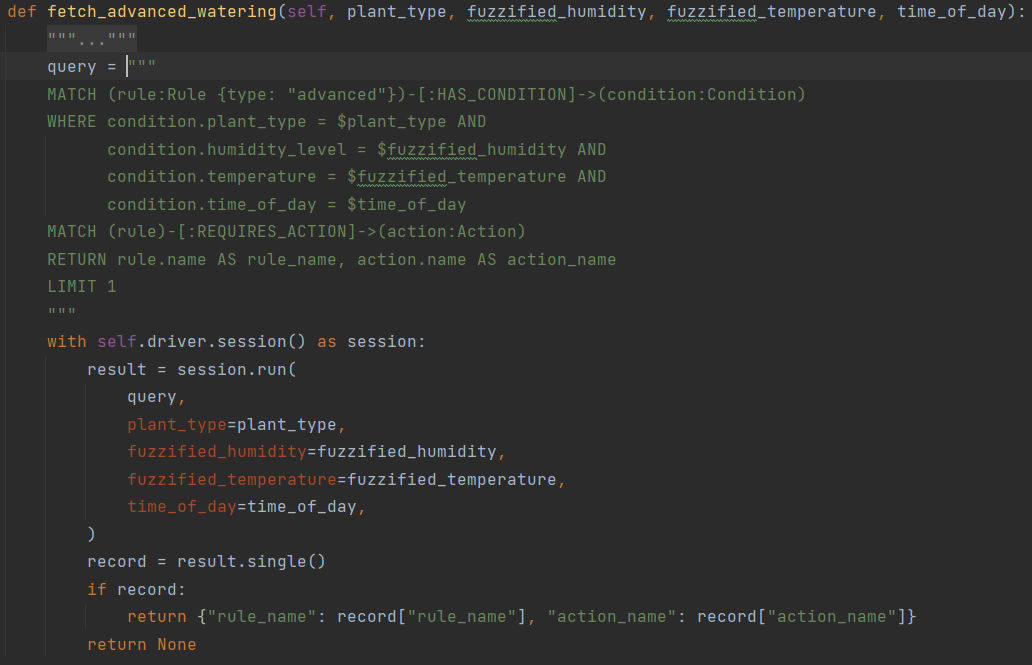
Файл rule\_engine.py будет являться связующим между симуляцией и базой данных.

Функция получает на вход температуру и влажность в численном формате, время дня и тип растения. Сначала, пользуюсь нечёткой логикой (треугольная функция принадлежности), он преобразует температуру и влажность в строке, после вызывает метод из класса Neo4j, и если правило нашлось, то выводит его результат, а самой симуляции отдаёт дефузиффицированное значение процента влажности, который нужно добавить растению.



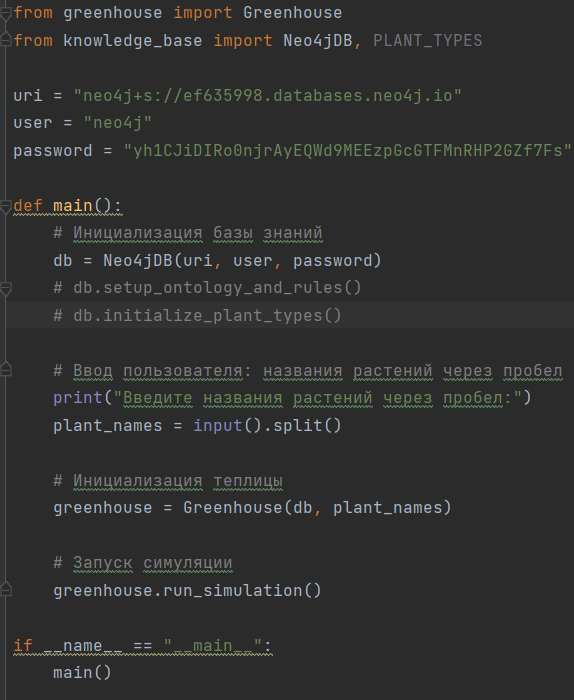


По запросу Cypher программа ищет правило типа «basic», который связан с условием нескольких свойств (влажность и тип растения), затем ищет связанное с правилом действие и возвращает имя правила с действием.

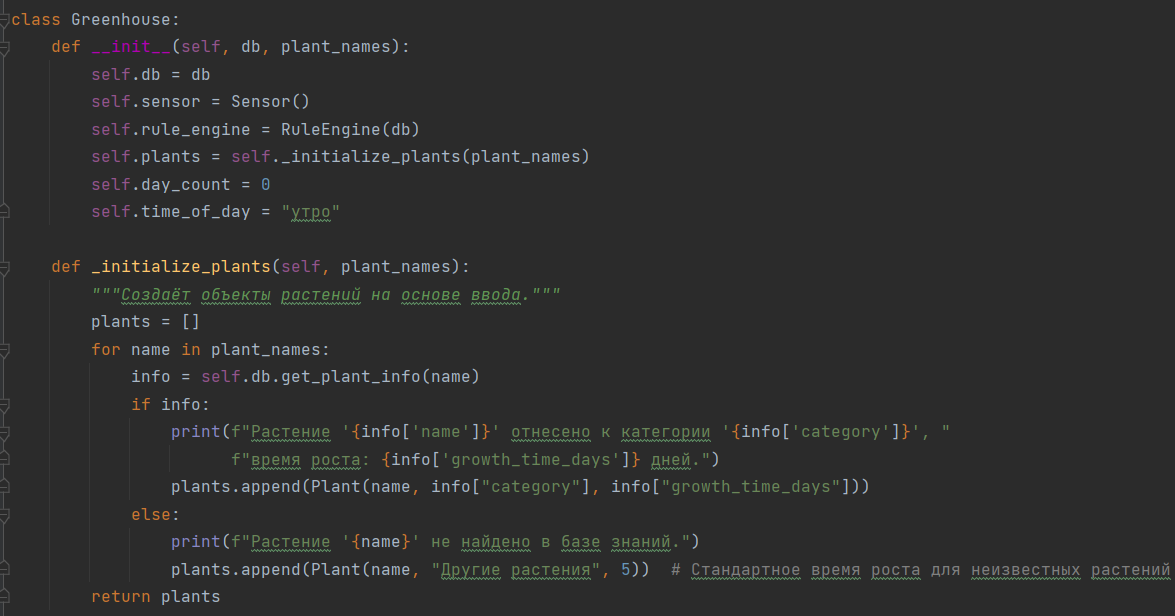


Логика поиска продвинутого правила почти такая же, только условий стало больше.

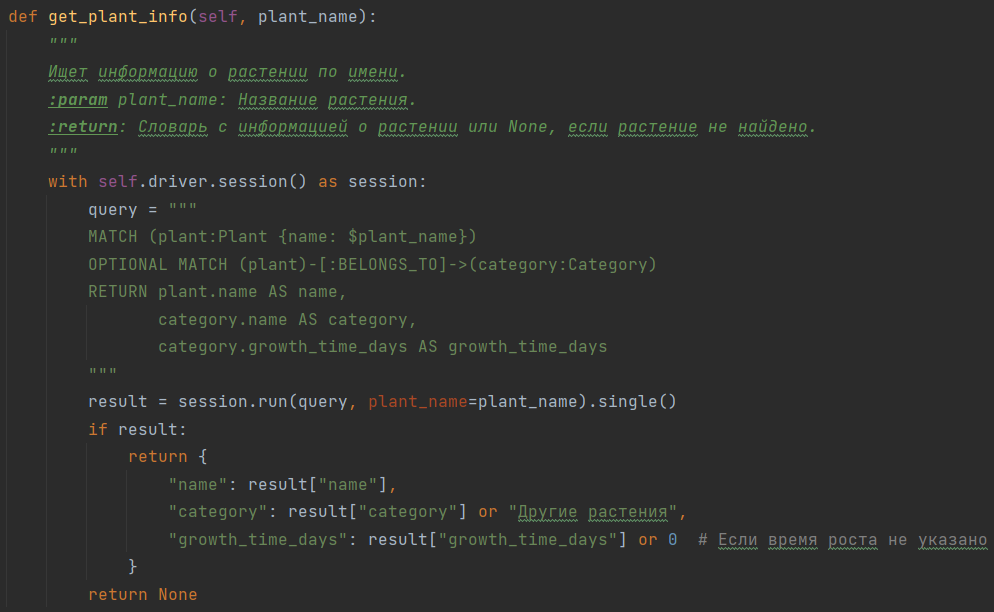
1. Создание симуляции.



Изначально программа требует от пользователя ввести названия растений через пробел – это будут ячейки теплицы. Каждая ячейка будет поливаться отдельно.



Класс Greenhouse изначально определяет тип растения, поэтому он сразу обращается к классу Neo4j функции get\_plant\_info:



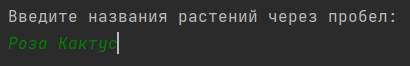
Сначала проводится поиск растения в базе. Если такого нет, проводится поиск связи растения с типом, после производится возврат имени растения, типа растения и времени роста.

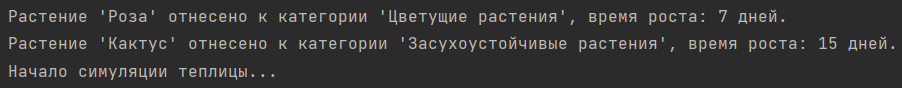
После определения типов растения производится основной цикл.

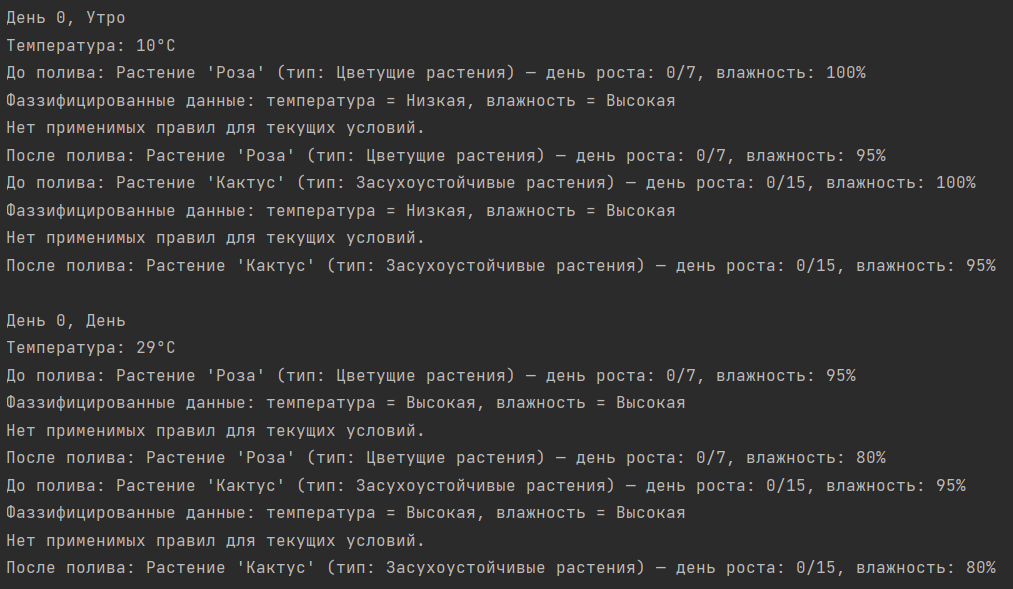
1. Случайно генерируется температура
2. Обновляется влажность растения (понижается от температуры)
3. Проверяются правила полива.
4. При наличии активных правил производится полив
5. Проверка на существование растений
6. Смена времени – отдельная функция, которая меняют ротации утра, дня, вечера и ночи, считает дни и считает дни растения. При этом происходит проверка, не выросло ли растение. В положительном условии выводится сообщение о том, что растение выросло, и он уходит из ячейки теплицы

Когда все растения вырастут – программа завершит свою работу.

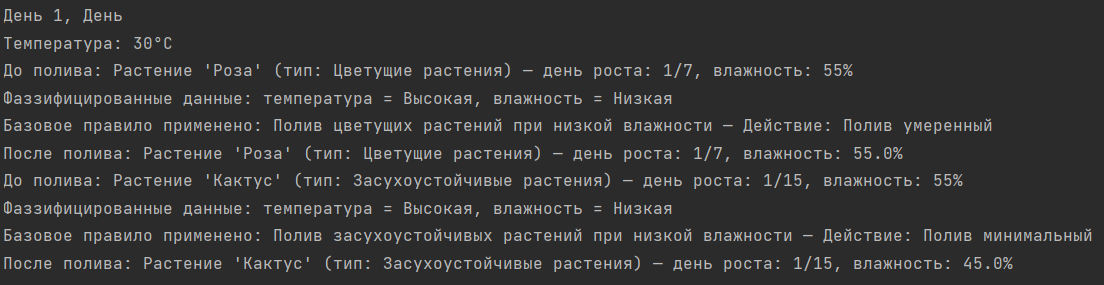
1. Тестирование



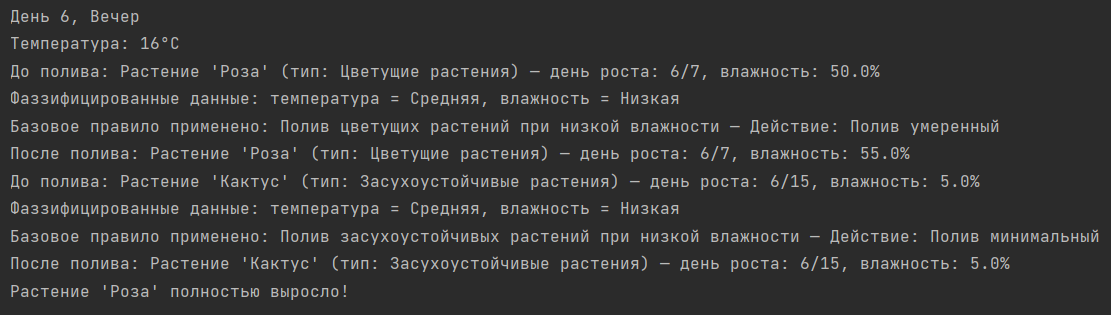




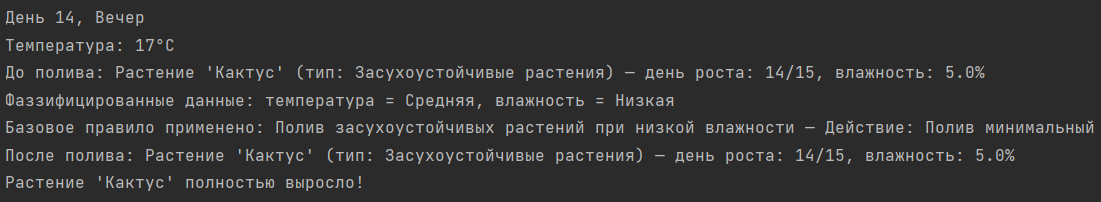
Применение правил полива произошло в первый раз днём 1-го дня



На вечер 6-го дня вывелось сообщение о том, что «Роза» выросла (это произошло после смены вечера на ночь):



На вечер 14-го дня вырос кактус, и теплица завершила свою работу.



**Вывод**

В ходе лабораторной работы была разработана система управления для автоматизации технического объекта. Для хранения базы правил и онтологии использовалась Neo4j. Были спроектированы логическая модель управления, настроены правила, реализованы фаззификация для обработки условий и дефаззификация для формирования управляющих инструкций. Также создан минимально рабочий симулятор предметной области с дискретным управлением для проверки работы системы.